J2EE 어플리케이션 모델 기반의 컴포넌트 저장소 구현

안성아[†] · 최희석[††] · 염근혁[†††]

요 약

현재는 소프트웨어의 재사용성, 신뢰성, 확장성을 높이기 위한 해결책으로서 컴포넌트 기반 소프트웨어 개발(Component Based Software Development)을 따르고 있다. 이를 지원하기 위해서는 개발된 컴포넌트들이 컴포넌트 저장소를 통하여 소프트웨어 개발 시 재사용되어야 한다. 그러나, 전통적인 중앙 집중식 저장소로서는 사용자 수의 제약 뿐만 아니라 제공하는 컴포넌트들도 사용자 인터페이스 등과 같은 클라이언트측 컴포넌트들이 대부분이었으므로 컴포넌트 시장의 활성화를 가져오지 못하였다.

최근에는 인터넷 기술과 함께 EJB, DCOM 등과 같은 컴포넌트 기술의 발달에 힘입어 특정 영역(Domain)에서 재사용되는 서버측 컴포넌트들이 많이 등장하였다. 따라서 컴포넌트 저장소는 도메인 정보를 기반으로 하여 컴포넌트들을 분류하여 제공하며, 많은 사용자들이 안정적으로 사용할 수 있도록 하기 위하여 웹 기반의 서비스를 제공해야 한다.

본 논문에서는 어플리케이션 개발자들에게 요구사항에 맞는 컴포넌트를 검색하여 추출하기 위한 서비스를 제공하며, 웹 환경에서 안정적이고 신뢰성 있는 서비스를 제공하기 위하여 컴포넌트 저장소의 요구사항 분석을 바탕으로 J2EE(Java 2 Enterprise Edition) 어플리케이션 모델 기반의 컴포넌트 저장소를 제시한다.

An Implementation of a Component Repository based on J2EE Application Model

Sungah Ahn[†], Heeseok Choi[††] and Keunhyuk Yeom[†††]

ABSTRACT

Now we are following component-based software development as a solution to improve the reusability, reliability, and extensibility of the software. To support these, the developed components must be reused through component repository at software development. However, traditional centralized repositories couldn't activate component markets since not only the number of users were restricted but also most of the provided components were those on clients like user interface and so on.

Recently, a lot of server-side components, reused in a specific domain, have appeared thanks to the development of both Internet technology and component technology such as EJB, DCOM, etc. Therefore, component repositories, based on domain information, classify and provide components and must offer web-based service for many users to use it stably.

This study introduces the implementation of a component repository based on J2EE application model. It provides application developers with the service to search and extract appropriate components, and supplies them reliable services in web environment.

Key words: 컴포넌트 저장소, J2EE, EJB

본 연구는 한국과학재단 목적기초연구(R02-2000-00276)지원으로 수행되었음.

[†] 부산대학교 컴퓨터공학과 석사과정

[††] 부산대학교 컴퓨터공학과 박사과정

[†††] 정회원, 부산대학교 컴퓨터공학과 조교수

1. 서 론

최근 복잡 다양해지는 소프트웨어의 품질 향상과 시기 적절한 소프트웨어 생산, 요구사항 변화에 대한 효과적 대응을 위하여, 소프트웨어를 재사용 가능한 컴포넌트 형태로 개발하는 기술이 빠르게 발전하고 있다. 이러한 컴포넌트 기반 소프트웨어 개발(Component Based Software Development)이 점차 보편화 되어감에 따라 컴포넌트의 개발과 컴포넌트의 획득을 위해 컴포넌트 유통체계의 필요성이 크게 대두되고 있다. 따라서 컴포넌트를 통합 관리할 수 있는 컴포넌트 저장소(Component Repository)의 개발이 필연적이다.

초기의 컴포넌트 저장소는 중앙 집중식 데이터베이스 형태로 개발되어 컴포넌트의 재사용을 지원하였다. 그러나 이러한 형태의 컴포넌트 저장소에 접근하기 위해서는 특정 클라이언트 프로그램을 설치해야 하고, 다수의 사용자가 컴포넌트를 등록 및 획득하는 데 있어서 데이터베이스에 직접 접근해야 하는 불편함과 위험성이 존재하였다. 최근 인터넷의 급속한 발전과 더불어 CGI나 Servlet과 같은 웹 기반 어플리케이션 개발 기술에 힘입어 다수의 컴포넌트 사용자가 다양한 환경에서 컴포넌트 저장소를 활용할 수 있는 웹 기반의 컴포넌트 저장소 형태로 발전하고 있다. 웹 기반의 컴포넌트 저장소는 컴포넌트의 등록, 저장, 획득, 유지 관리 등의 기능성과 함께 트랜잭션 처리에 있어서의 신뢰성과 안정성 보장, 다수 사용자에 의한 동시 접근을 원활하게 지원해야 하는 비기능성을 만족시켜야 한다. 또한 컴포넌트 저장소는 컴포넌트의 변동 및 관리 정책의 변화 등에 유연하게 대처해야 하고 기능의 확장이 용이하여야 한다. 그러나 기존에 사용되어진 CGI나 Servlet 등과 같은 웹기반 어플리케이션 개발 기술만으로는 이러한 요구 사항들을 충족시키기가 어렵다.

본 논문에서는 컴포넌트 기반의 웹 어플리케이션을 개발하는 데 적합한 모델로 고려되고 있는 선 마이크로시스템즈(Sun Microsystems Inc.)의 J2EE (Java 2 Platform Enterprise Edition) 어플리케이션 모델(J2EE application model: BluePrints)[1,2]을 바탕으로 하여 웹 기반 컴포넌트 저장소를 설계 및 구현하였다. J2EE는 컴포넌트 기반의 다계층 분산 어플리케이션 모델로서, 복잡한 분산 객체 프레임워크에 대한 작성 없이 확장성과 신뢰성이 높은 어플리케이션 작성을 지원한다. 또한 J2EE 어플리케이션 모델에서 비즈니스 로직(business logic)과 데이터 처리를 담당하는 EJB(Enterpise JavaBeans)는 데이터 접근, 동시성, 트랜잭션 등과 같은 복잡한 문제를 효율적으로 해결해 준다[2]. 따라서 웹 기반 컴포넌트 저장소를 개발하기 위하여 저장소의 기능성과 비기능성을 명확하게 정의한 후, J2EE 어플리케이션 모델을 바탕으로 한 웹 기반 컴포넌트 저장소를 설계하였고, 이를 위하여 J2EE 어플리케이션 모델과 같은 다계층 분산 어플리케이션 모델을 고려한 개발 방법인 분산 컴포넌트 기반의 소프트웨어 분석 및 설계 방법[3,4]과 UML(Unified Modeling Language)[5]을 사용하였다. 그리고 이러한 설계를 토대로 J2EE 환경이 제공하는 서비스를 활용하여 웹 기반의 컴포넌트 저장소를 구현하였다. 개발된 컴포넌트 저장소는 요구되는 컴포넌트 등록, 저장, 획득 등의 기능을 제공하고, 신뢰성과 안정성, 확장성, 유지 관리의 용이성을 충족시킨다.

2. 관련 연구

2.1 컴포넌트 저장소

컴포넌트 저장소는 컴포넌트를 등록하고, 구입 및 유통 등의 컴포넌트에 대한 통합 관리를 위한 시스템이다. 그러나 지금까지 컴포넌트 저장소에 관련된 연구는 컴포넌트를 효율적으로 추출하기 위한 컴포넌트 명세와 분류에 대한 내용이 주류를 이루고 있고 웹 환경에서 지원되는 컴포넌트 저장소의 구현에 관련된 연구는 많이 부족한 실정이다. 현재 컴포넌트 저장소 구현에 관련된 연구로는 미국 카네기멜론 대학의 SEI(Software Engineering Institute)에서 웹 기반의 검색엔진을 이용하여 JavaBean, ActiveX, CORBA, EJB 등으로 분류된 컴포넌트들을 제공하는 Agora[6]라는 컴포넌트 저장소의 프로토타입이 있다. 그리고 상용적으로 운용되고 있는 컴포넌트 저장소로는 대표적으로 영국의 ComponentSource[7]와 미국의 Imagicom[8] 등을 들 수 있다.

2.2 분산 컴포넌트 기반의 소프트웨어 분석 및 설계 방법[3,4]

분산 컴포넌트 기반의 소프트웨어 분석 및 설계

방법은 요구사항 추출단계, 분석 단계, 설계 단계로 구성된다. 요구사항 추출 단계에서는 소프트웨어의 외부적인 측면의 기능을 중심으로 요구사항을 추출한다. 분석 단계에서는 추출된 요구사항을 바탕으로 하여 소프트웨어의 내부적인 측면의 기능을 중심으로 분석한다. 설계 단계에서는 기능 중심의 분석으로부터 분산 컴포넌트를 추출하고 다양한 분산 측면을 고려한다. 각 단계별로 자세히 살펴보면 다음과 같다.

2.3.1 요구사항 추출

소프트웨어의 요구사항을 추출하여 use case와 actor로 구성되는 기능 중심의 use case 모델을 생성한다. 먼저 기능적 요구사항을 중심으로 기능과 관련 있는 다양한 actor를 추출하고 기능에 해당되는 use case를 추출한다. use case 모델을 형성하기 위하여 추출한 use case를 중심으로 actor를 분류하여 association을 형성하고 use case간의 관계를 일반화(generalization), 확장(extend), 포함(include)관계로 분류한다. 생성된 use case 모델의 각 use case에 대하여 statechart diagram, activity diagram, collaboration diagram을 이용하여 use case description을 표현한다.

2.2.2 분석

요구사항으로부터 생성한 use case 모델을 이용하여 분석 모델을 생성한다. 먼저 기능 중심의 독립적인 단위인 analysis package를 추출하고, 추출된 analysis package간의 의존(dependency) 관계를 형성한다. 또한 analysis package의 세부적인 구성요소인 analysis class를 추출한다.

2.2.3 설계

Use case 모델로부터 기능적인 측면을 중심으로 생성한 분석 모델을 바탕으로 설계 모델을 생성한다. 먼저 분산 컴포넌트를 생성하는 작업을 하고 추출된 분산 컴포넌트에 해당되는 analysis class를 바탕으로 분산 이슈를 고려하여 분산 컴포넌트와 클래스를 설계한다.

2.3 J2EE[1, 2, 9, 10, 11, 12]

J2EE는 현대적이고 확장된, 전자 비즈니스 지향적인 엔터프라이즈 어플리케이션 시스템의 엄격한 요건을 지원하도록 설계되었기 때문에, 컴포넌트에 기반하고 서버 중심적인 다계층 어플리케이션 구조를 제공한다[13].

2.3.1 J2EE 구조

J2EE는 웹 개발을 포함한 모든 엔터프라이즈 급 개발을 위한 광범위한 기술로서 그 구조는 그림 1과 같다.

그림 1에서 나타나는 J2EE의 구성 요소들은 크게 표 1과 같이 세 종류로 나누어 볼 수 있다.

표 1에서 컴포넌트 기술은 J2EE 어플리케이션의 비즈니스 로직과 프리젠테이션 로직(presentation logic)을 구성하는 요소들을 의미하며, J2EE 어플리케이션 개발자가 실제로 개발해야 할 주체들이다. 서비스 기술은 J2EE 어플리케이션이 기존 시스템에서 제공하는 서비스를 활용할 수 있도록 하는 API들이다. 통신 기술에 속하는 요소들은 객체 간 통신을 위한 것이거나, 메시지 관련 어플리케이션을 개발하기 위해 사용되는 것들이다.

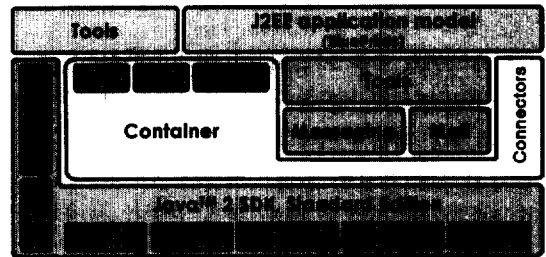



그림 1. J2EE 구조 [14]

표 1. J2EE 요소 기술의 분류 [2]

구 분	설 명	요소 기술
컴포넌트	개발자가 개발의 주체로 삼는 애플리케이션을 구성하는 컴포넌트	Servlet, JSP, EJB
서 비 스	데이터베이스 제어, 이름 및 디렉토리 서비스, 트랜잭션 처리 등을 위한 컴포넌트가 사용하는 API	JDBC, JNDI, JTA
통 신	메시지 관련 어플리케이션 작성을 위한 혹은 프로토콜에 관련된 API	JMS, JavaMail, JAF, RMI-IIOP

2.3.2 J2EE 어플리케이션 모델 (BluePrints)

J2EE는 엔터프라이즈, 다계층 어플리케이션을 개발하기 위한 다계층 분산 어플리케이션 모델을 제공하며 그 구조는 그림 2와 같다.

그림 2는 J2EE의 어플리케이션 모델로서, 어플리케이션이 3계층, 즉 클라이언트 층, 중간 층, EIS 층으로 나누어져 있으므로 각 부분이 다른 장치(device)에서 동작 가능하다는 것을 의미한다. 각 계층에 대하여 살펴보면 다음과 같다.

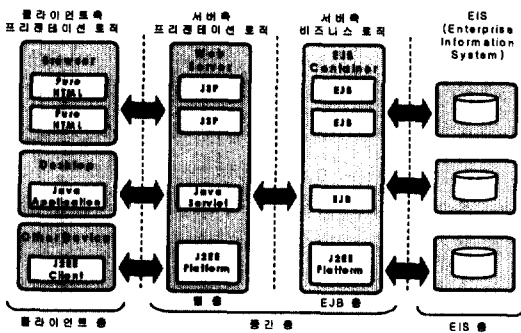



그림 2. J2EE 어플리케이션 모델 (BluePrints) (15)

① 클라이언트 층 (Client Tier)

클라이언트 층은 사용자를 위한 인터페이스를 담당하는 부분이다. 웹 층이나 EJB 층의 컴포넌트를 사용하거나 직접 EIS에 접근하여 결과를 사용자에게 전달하는 역할을 한다.

② 중간 층 (Middle Tier)

중간 층은 크게 웹 층과 EJB 층으로 나누어진다. 웹 층은 Servlet과 JSP 두 종류의 컴포넌트로 구성되며, 프리젠테이션 로직을 담당한다. EJB 층은 J2EE 어플리케이션 아키텍처 내에서 가장 중요한 부분으로서 EJB를 구동하기 위하여 EJB 컨테이너가 존재하며, 비즈니스 로직을 담당한다.

③ EIS 층 (EIS Tier)

EIS는 기존의 정보 시스템을 말한다. EIS의 대표적인 것으로는 데이터베이스나 ERP(Enterprise Resource Planning)를 꼽을 수 있다. 비즈니스에 필요한 정보들은 이미 구축된 EIS 내에 저장되어 있으므로, J2EE 어플리케이션은 EIS에 접근 가능해야 한다. J2EE 어플리케이션 아키텍처는 EIS를 사용하여 영구적인 데이터 저장을 전제로 하기 때문에, 데이터베이스 시스템 등의 사용은 필수적이다.

2.4 EJB (2, 9, 10, 12, 16, 17)

EJB는 엔터프라이즈 환경을 위한 서버 측 분산객체 컴포넌트이다. J2EE 어플리케이션 내에서 EJB는 비즈니스 로직을 모두 담당한다. 사용자에게 보여주는 인터페이스 부분을 제외한 모든 데이터 처리와 비즈니스 로직이 EJB로 구성된다. EJB에는 세션 빈(Session Bean)과 엔티티 빈(Entity Bean)의 두 가지 유형이 있다.

2.4.1 세션 빈

세션 빈은 클라이언트 어플리케이션의 확장으로, 비즈니스 로직을 처리하는 역할을 한다. 데이터베이스에 많은 영향을 미치지만, 데이터베이스 내의 공유된 데이터를 표현하는 것이 아니라, 공유된 데이터에 엑세스하는 역할을 한다. 퍼시스턴트(persistent)한 상태를 가지지 않는다.

① 상태유지 세션 빈

오직 하나의 클라이언트에 할당되고 그 클라이언트와 관련된 상태를 유지한다. 비즈니스 로직과 클라이언트의 상태를 캡슐화하여 이를 서버에 전달 할 수 있도록 해주므로 클라이언트 어플리케이션이 더 가벼워지고, 시스템은 관리하기가 용이해진다.

② 무상태 세션 빈

아주 일반적이고 재사용 가능한 서비스를 수행한다. 하나의 클라이언트에 할당되지 않기 때문에 클라이언트의 상태를 유지하지 않는다. 메소드가 알아야 하는 모든 사항은 메소드의 매개변수를 통해 전달된다.

2.4.2 엔티티 빈

엔티티 빈은 현실세계의 상태와 동작을 표현한다. 퍼시스턴트한 상태를 가지며 퍼시스턴스를 관리하는 방법에 따라 컨테이너 관리 빈과(Container-managed bean) 빈 관리 빈(Bean-managed bean)으로 구분된다.

① 컨테이너 관리 빈

퍼시스턴스가 EJB 컨테이너에 의해서 자동으로 관리된다. 컨테이너는 빈 인스턴스의 필드가 어떻게

데이터베이스에 대응되어야 하는지를 알고, 데이터베이스 내의 엔티티와 관련된 데이터를 추가, 변경, 삭제하는 것을 자동으로 관리한다. 따라서 빈이 그 상태를 저장하는 데에 사용하는 데이터베이스와 독립적으로 정의될 수 있으므로 응용 프로그램간의 빈의 재사용성과 유연성을 높일 수 있다.

② 빈 관리 빈

퍼시스턴스를 관리하는 모든 작업을 개발자가 명시적으로 한다. 빈 개발자는 데이터베이스를 다루는 코드를 반드시 사용하여야 한다.

2.4.3 배치 기술자 (Deployment Descriptor 또는 DD)

배치 기술자는 EJB-jar 파일 생산자와 소비자 사이 간 계약의 일부로 구조적 정보, 조립 정보 등이 포함된다. 빈 제공자에 의해 만들어진 하나의 EJB-jar 파일은 하나 이상의 엔터프라이즈 빈들을 포함하며, 보통 응용프로그램 조립에 관련된 지시사항들은 포함하지 않는다.

3. 웹 기반의 컴포넌트 저장소 설계

웹 기반의 컴포넌트 구현에 있어서 J2EE 환경에서 제공해주는 여러 가지 장점을 잘 이용하기 위해서는 컴포넌트 저장소의 요구사항 분석, 설계 단계에서부터 J2EE 어플리케이션 모델이 고려되어야 한다. 즉, J2EE 어플리케이션 모델은 컴포넌트 기반이므로 컴포넌트 기반의 개발 방법을 사용한 설계가 필요하다. 그리고 J2EE 어플리케이션 모델은 다계층, 즉 세 개의 층으로 나누어지며, 이 층들은 클라이언트 층, 웹 층과 EJB 층이 합쳐진 중간 층, 그리고 EIS 층으로 나누어지고, 각 층들은 프리젠테이션 로직, 비즈니스 로직, 데이터베이스 중 하나의 역할을 수행하게 된다. 그러므로 추출된 컴포넌트의 배치에 있어서도 이러한 사항을 잘 따를 수 있는 설계 방법이 요구되어진다.

따라서 본 논문에서는 J2EE 어플리케이션 모델과 같은 컴포넌트 기반이며 다계층 분산 어플리케이션의 특징을 잘 고려한 설계 방법인 분산 컴포넌트 기반의 소프트웨어 분석 및 설계 방법[3,4]을 사용한다. 즉, J2EE 어플리케이션 모델의 특징을 잘 만족시키기 위해 먼저, 분석 및 설계 단계에서 웹 기반의 컴포넌트 저장소의 기능적 요구사항으로부터 시스템을 컴포넌트화하고 각 컴포넌트의 기능과 역할에 따라 적절한 층에 배치한 후, 구현 단계에서는 각 층별로 사용되어지는 J2EE의 요소 기술을 바탕으로 구현하는 과정이 진행된다.

요구사항 추출 단계에서는 기능적 요구사항을 분석하여 use case와 actor를 추출하고, 분석 단계에서는 각 use case로부터 analysis package를 통해 analysis class를 추출한다. 그리고 설계 단계에서는 추출된 analysis class들을 바탕으로 컴포넌트를 추출하고 기능과 역할에 따라 J2EE 어플리케이션 모델의 각 층에 배치하는 과정을 진행한다. 본 논문에서는 개발 진행 과정을 컴포넌트 저장소의 기능 중 컴포넌트에 관련한 예를 바탕으로 설명하고자 한다.

3.1 요구사항 추출

컴포넌트 저장소는 컴포넌트를 등록하고자 하는 측면과 구입하고자 하는 측면 즉, 컴포넌트 개발자와 컴포넌트 구매자(어플리케이션 개발자)의 2가지 측면으로 고려되어질 수 있고, 각 측면에 따라 다음과 같은 서비스를 제공해야 한다.

◆ 컴포넌트 개발자에게 제공하는 서비스

컴포넌트 개발자에 의해 개발된 컴포넌트들을 유통하기 위한 서비스로서 컴포넌트 개발자로 요구되는 모든 사항을 등록한 뒤, 부여받은 아이디를 통하여 사용자 인증을 거친 후에 컴포넌트를 등록하게 된다. 컴포넌트 등록 시에는 컴포넌트 저장소에서 요구하는 모든 메타정보에 대한 내용을 처리하여 저장해야한다. 또한 등록된 컴포넌트에 대한 정보를 업데이트할 경우 컴포넌트 사용자에게 모든 정보를 업데이트해야 하며 개발자가 등록한 컴포넌트에 대한 조회가 가능해야 한다.

◆ 컴포넌트 구매자(어플리케이션 개발자)에게 제공하는 서비스

컴포넌트 구매자는 컴포넌트를 구매하기 위해 구매자로 등록한 뒤 사용자 인증을 통하여 요구 사항에 맞는 컴포넌트를 추출할 수 있도록 컴포넌트 메타정보에 의한 키워드 검색과 도메인 분류에 의한 검색을 지원한다. 또한 구매한 컴포넌트에 대한 업데이트 정보를 자동통보 받아야 하며, 거래한 컴포넌트 내역에 대한 조회기능을 제공해야 한다.

위와 같이 제공되는 서비스는 컴포넌트 저장소에 대한 기능적 요구사항들이다. 이 외에도 웹 기반의 컴포넌트 저장소를 구축하기 위해서는 많은 사용자들이 동시 접속 시에 안정된 서비스를 제공해 줄 수 있는 안정성, 트랜잭션 처리에 있어서의 신뢰성, 유지 관리의 효율성, 컴포넌트의 변동 및 관리 정책의 변화에 잘 대처할 수 있는 유연성, 추가되는 기능에 대한 확장성 등의 비기능적 요구사항들이 존재한다.

컴포넌트 저장소에 대한 기능적 요구사항들을 바탕으로 시스템에 참여하는 actor와 use case를 추출하면 그림 3과 같다.

actor로서는 컴포넌트 개발자와 컴포넌트 구매자

*Actor 추출
- 컴포넌트 개발자
  + 컴포넌트 저장소에 컴포넌트를 등록하는 일(업로드)을 수행한다.
- 컴포넌트 구매자
  + 컴포넌트 메타 정보를 통하여 컴포넌트를 구입(다운로드)한다.

*Use case 추출

컴포넌트 개발자 측면
- 신규 컴포넌트 개발자 등록
  + 컴포넌트 개발자 등록 폼을 작성한다.
  + 컴포넌트 개발자 아이디 및 패스워드 발부
- 컴포넌트 개발자 로긴
  + 아이디와 패스워드를 통하여 컴포넌트 개발자로 접속한다.
- 컴포넌트 개발자 정보 업데이트
  + 컴포넌트 개발자에 대한 정보 변경시 업데이트를 한다.
- 컴포넌트 등록
  + 컴포넌트 개발자의 인증을 거친 후 컴포넌트를 등록한다.
  + 컴포넌트 등록 후 컴포넌트에 대한 고유 등록번호를 받는다.
- 컴포넌트 정보 업데이트
  + 등록번호를 통하여 컴포넌트 변경사항에 대한 업데이트를 한다.
- 컴포넌트 등록 내역 조회
  + 지금까지 등록하였던 컴포넌트의 이름, 등록 날짜, 거래 회수, 등록 번호 등을 알 수 있다.

컴포넌트 구매자 측면
- 신규 컴포넌트 구매자 등록
  + 컴포넌트 구매자 등록 폼을 작성한다.
  + 컴포넌트 구매자 아이디 및 패스워드 발부
- 컴포넌트 구매자 로긴
  + 아이디와 패스워드를 통하여 컴포넌트 개발자로 접속한다.
- 컴포넌트 구매자 정보 업데이트
  + 컴포넌트 구매자에 대한 정보 변경시 업데이트를 한다.
- 문자기반 컴포넌트 검색
  + 컴포넌트를 키워드를 통하여 찾는다.
  + 검색 결과의 컴포넌트들을 테이블로 보여준다.
  + 구매자가 컴포넌트에 대한 설명, 명세, 모델링의 정보를 볼 수 있다.
  + 컴포넌트를 결정 후에 장바구니에 넣는다.
- 메타정보기반 컴포넌트 검색
  + 컴포넌트의 분류 화면을 보여준다.
  + 컴포넌트 카탈로그를 통하여 컴포넌트를 찾는다.
  + 구매자가 컴포넌트에 대한 설명, 명세, 모델링의 정보를 볼 수 있다.
  + 컴포넌트를 찾은 후에는 장바구니에 넣는다.
- 장바구니 보기
  + 장바구니에 존재하는 컴포넌트를 보여준다.
- 주문 절차
  + 주문시 필요한 사항을 입력한다.
- 컴포넌트 구입 내역 조회
  + 지금까지 구입한 컴포넌트들의 거래정보를 알 수 있다.

그림 3. actor와 use case 추출

를 추출하였고, 각각의 actor들과 관계를 가지는 14개의 use case들을 추출하였다. 이를 바탕으로 UML [18] 표기법을 이용하여 use case diagram을 작성하면 그림 4와 같다.

그림 4에서는 컴포넌트 개발자와 컴포넌트 구매자의 측면에서 컴포넌트 저장소에 대한 use case diagram을 나타내었다. 그리고 각각의 use case에 대하여 actor, 목적, 선행조건, 이벤트의 흐름, 예외사항, 종료 조건과 같은 내용을 기본으로 하여 use case 명세서를 작성하였다. 그림 5는 use case 명세서에 대한 예이다.

그림 5의 예는 기능적 요구사항에서 추출된 use case들 중에서 컴포넌트에 관련된 use case들을 선택하여 그 use case에 대한 명세를 나타낸 것이다.

3.2 분석 및 설계

분석 및 설계 단계에서는 각 use case로부터 analysis package를 통해 analysis class를 추출한다. 추출된 analysis class들을 바탕으로 컴포넌트를 추출하고 기능과 역할에 따라 J2EE 어플리케이션 모델의 각 층에 배치하는 과정을 진행한다. 여기서 진행 사항을 설명하기 위해 이전 단계의 그림 5의 예로서 사용된 컴포넌트 관련 use case들을 사용하였다. 이러한 use case들로부터 analysis package를 통해 analysis class를 추출한다. 추출된 analysis class를 살펴

그림 4. use case diagram

- 컴포넌트 등록
  + Actor
    컴포넌트 개발자
  + 목적
    컴포넌트를 저장소에 등록한다.
  + 선행조건
    컴포넌트 등록 메뉴를 선택한다.
  + 이벤트 흐름
    1. 등록 시 기입해야 할 사항을 보여준다.
    2. 등록자가 등록 완료버튼을 누르면 기입된 사항을 확인한다.
    3. 컴포넌트 개발자 로긴과정을 거친다.
    4. 기입된 사항들을 바탕으로 DB를 검색하여 기 등록 컴포넌트인지 확인한다.
    5. 등록된 컴포넌트에 고유 등록번호를 부여한다.
    6. 기입된 컴포넌트 정보를 기준에 따라 DB에 저장한다.
    7. 등록된 사항(컴포넌트 고유 등록번호 포함)을 보여준다.
  + 예외 사항
    2a. 컴포넌트 등록 필수사항이 모두 기입되지 않은 경우
      2a1. 에러 메시지를 발생시킨다.
      2a2. 이벤트 흐름 1번으로 돌아간다.
    4a. 기 등록 컴포넌트인 경우
      4a1. 에러 메시지를 발생시킨다.
      4a2. 비정상 종료 (등록 실패)
  + 종료 조건
    등록 완료 확인 버튼을 누른다.

그림 5. (a) 컴포넌트 등록 use case 명세

- 메타정보기반 컴포넌트 검색
  + Actor
    컴포넌트 구매자
  + 목적
    메타정보 기반으로 컴포넌트를 검색한다.
  + 선행조건
    컴포넌트 검색 메뉴를 선택한다.
  + 이벤트 흐름
    1. 컴포넌트 분류 화면을 보여준다.
    2. 컴포넌트 카탈로그를 통해 컴포넌트를 찾는다.
    3. 상세 정보 버튼을 선택하면 컴포넌트에 대한 상세 정보를 보여준다.
    4. 컴포넌트 결정 후 장바구니에 넣는다.
  + 예외 사항
    4a. 컴포넌트 구매자 로긴이 되어 있지 않은 경우
      4a1. 컴포넌트 구매자 로긴 과정을 수행한다.
      4a2. 장바구니에 넣는다.
  + 종료 조건

그림 5. (b) 메타정보기반 컴포넌트 검색 use case 명세

- 컴포넌트 정보 업데이트
  + Actor
    컴포넌트 개발자
  + 목적
    저장소에 등록되어 있는 컴포넌트 정보를 업데이트한다.
  + 선행조건
    로긴 상태이고 컴포넌트 정보 업데이트 메뉴를 선택한다.
  + 이벤트 흐름
    1. 컴포넌트 등록번호 입력 창을 보여준다.
    2. 입력된 등록번호를 사용하여 DB에 등록된 컴포넌트를 검색한다.
    3. DB에 등록된 사항을 보여준다.
    4. 업데이트 완료버튼을 누르면 수정된 사항을 확인한다.
    5. DB에 저장한다.
    6. 업데이트된 사항을 보여준다.
  + 예외 사항
    2a. 잘못된 등록번호일 경우
      2a1. 에러 메시지를 발생시킨다.
      2a2. 이벤트 흐름 1번으로 돌아간다.
    2b. 등록된 컴포넌트가 없을 경우
      2b1. 에러 메시지를 발생시킨다.
      2b2. 메뉴를 끝낸다.
    4a. 컴포넌트 등록 필수사항이 모두 기입되지 않은 경우
      4a1. 에러 메시지를 발생시킨다.
      4a2. 이벤트 흐름 1번으로 돌아간다.
  + 종료 조건
    업데이트 완료 확인 버튼을 누른다.

그림 5. (c) 컴포넌트 정보 업데이트 use case 명세

보면 그림 6과 같다.

그림 6의 (a)는 그림 5의 use case에서 추출된 analysis class들을 나타낸 analysis class diagram이고 (b)는 그림 5의 (a)에 명세된 컴포넌트 등록 use case로부터 추출된 analysis class들의 collaboration diagram을 나타낸 것이다. 이렇게 use case로부터 추출된 analysis class는 컴포넌트로 추출된다. 이때, 본 논문에서는 컴포넌트 추출 시에 analysis class의 기능적 유사성을 고려한다.

시스템이 컴포넌트화 되면 각각의 컴포넌트는 서

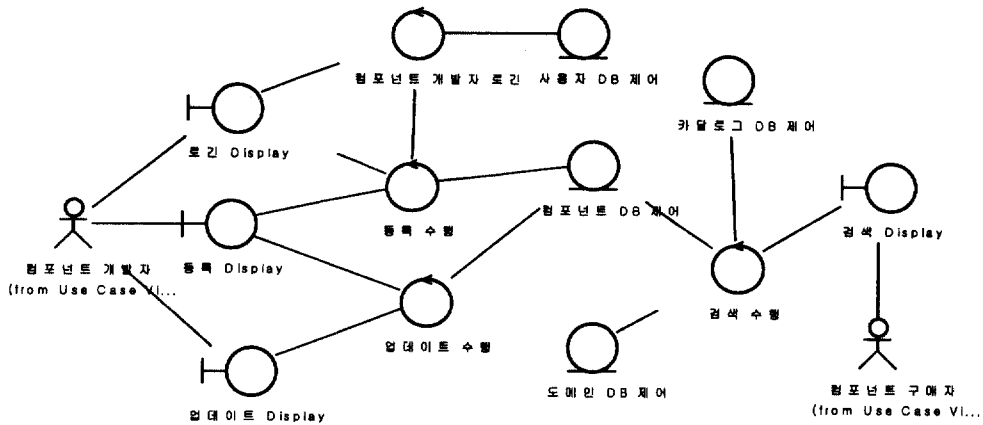



그림 6. (a) 컴포넌트에 관련된 use case들의 analysis class diagram

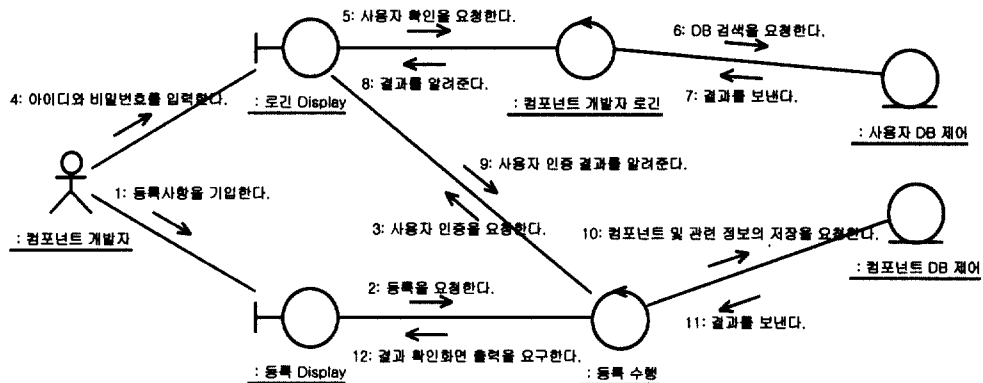



그림 6. (b) analysis class의 collaboration diagram의 예

로 다른 기능에 대해서 독립적이게 되므로 기능의 확장이나 변화 시에 서로 영향을 주지 않고 변경이 가능하게 된다. 따라서 시스템의 확장성이 높아지고 기능과 정책의 변화에 유연하게 대처 가능하며, 유지관리에도 효율적인 장점을 가지게 된다.

위의 그림 6의 (a)를 바탕으로 유사한 기능을 가진 analysis class를 살펴보면, 등록 Display class와 업데이트 Display class는 컴포넌트의 정보를 화면에 보여주는 역할을 하는 것으로 유사한 기능을 수행한다. 따라서 컴포넌트 정보를 화면에 Display하는 기능을 가지는 하나의 컴포넌트로 추출될 수 있다. 그리고 등록 수행 class와 업데이트 수행 class는 컴포넌트의 정보를 처리하는 기능을 수행하는 것으로서, 서로 유사한 기능을 가지기 때문에 하나의 컴포넌트로 추출될 수 있다. 컴포넌트 DB 제어 class, 사용자 DB 제어 class 등은 데이터베이스를 제어한다는 것은 같으나 서로 다른 데이터베이스를 제어할 뿐만 아니라 그 데이터의 처리에 있어서도 각기 판이하게 다른 과정을 수행하므로 하나의 컴포넌트로 추출되지 않고 각각 다른 컴포넌트로 추출된다. 그림 7은 analysis class로부터 컴포넌트를 추출한 것이다.

그림 7에서는 화면 출력과 관련이 있는 Login, FileUp, Searchs컴포넌트와 데이터베이스와 관련된 Catalogs, Components, Domains, User컴포넌트, 그리고 핵심 기능인 비즈니스 로직을 수행하는 Component Upload, UserHandle, Search컴포넌트, 이렇게 총 9개의 컴포넌트가 추출되었다.

추출된 컴포넌트는 그 역할에 따라 배치되는 층이 달라진다. 서버측 프리젠테이션 로직을 수행하는 컴포넌트는 중간 층에 배치되어지는데 그 중에서도 웹 층에 배치된다. 비즈니스 로직을 수행하는 EJB Bean 즉, Entity Bean과 Session Bean 또한 중간 층에 배

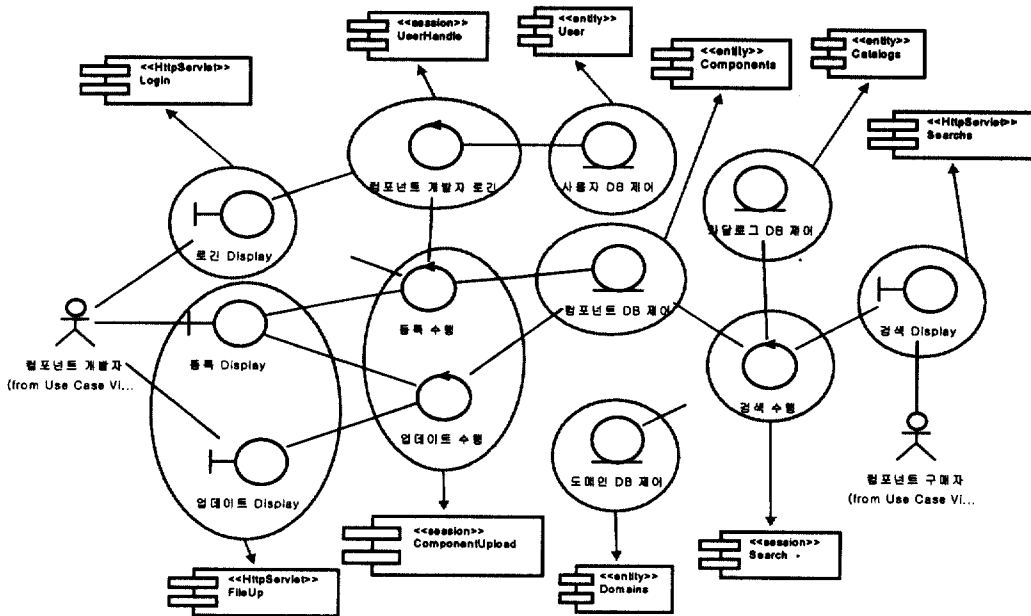



그림 7. 컴포넌트 추출

치되며 그 중에서도 EJB 층에 배치된다. 따라서 위의 그림 7에서 추출된 9개의 컴포넌트 중 화면 출력과 관련이 있는 Login, FileUp, Search컴포넌트는 서버측의 프리젠테이션 로직을 수행하므로 중간 층 중 웹 층에 배치된다. 그리고 데이터베이스와 관련된 Catalogs, Components, Domains, User컴포넌트와 핵심 기능인 비즈니스 로직을 수행하는 Component Upload, UserHandle, Search컴포넌트는 모두 중간 층의 EJB층에 배치된다.

클라이언트 층에는 클라이언트측 프리젠테이션 로직이 배치되는데, 본 논문에서는 클라이언트측 프리젠테이션 로직을 HTML로 구현하였고, 구현물로서 ComponentUpload.html, Login.html, Searchs.html 등이 있다. 그리고 EIS 층에는 데이터베이스가 배치되며 사용된 데이터베이스로는 컴포넌트 데이터베이스, 사용자 데이터베이스, 카달로그 데이터베이스, 도메인 데이터베이스 등이 있다. 그림 8은 그림 7의 컴포넌트 추출과정에서 추출된 컴포넌트들이 J2EE 어플리케이션 모델을 바탕으로 배치된 모습을 나타낸 것이다.

그림 8의 각 층들은 서로 다른 장치에 존재 가능하여 독립적이므로, 한 층의 컴포넌트를 변경하여도 다른 층에 영향을 주지 않는다. 따라서 이러한 J2EE 어플리케이션 모델의 특징을 고려하여 설계한 웹 기반의 컴포넌트 저장소는 컴포넌트의 변경이나 관리 정책상의 변화 시에 유연하게 대처 가능하고 시스템의 확장이 용이하며 유지, 관리 면에서도 효율성을 높일 수 있다.

4. 웹 기반의 컴포넌트 저장소 구현

컴포넌트 저장소의 구현은 분석 및 설계 단계에서 나온 결과물을 바탕으로 J2EE에서 제공하는 요소 기술을 사용하여 진행된다. J2EE에서 제공하는 요소 기술은 다양한 서비스를 제공하고 이러한 서비스를 이용하여 컴포넌트 저장소를 구현함으로써 웹 기반 컴포넌트 저장소의 많은 비기능적 요구사항을 충족시킬 수 있다.

J2EE에서 제공하는 다양한 요소 기술 중에서 본 논문의 웹 기반의 컴포넌트 저장소 구현 시에 사용되어진 요소 기술들은 그림 9와 같다.

그림 9에서 보는 바와 같이 클라이언트 측 프리젠테이션 로직을 담당하는 클라이언트 층의 구현 기술로는 HTML을 사용하였다. 그리고 중간 층 중 서버측 프리젠테이션 로직을 담당하는 웹 층은 비즈니스 로직의 결과물을 웹 환경에서 HTTP를 사용하여 웹

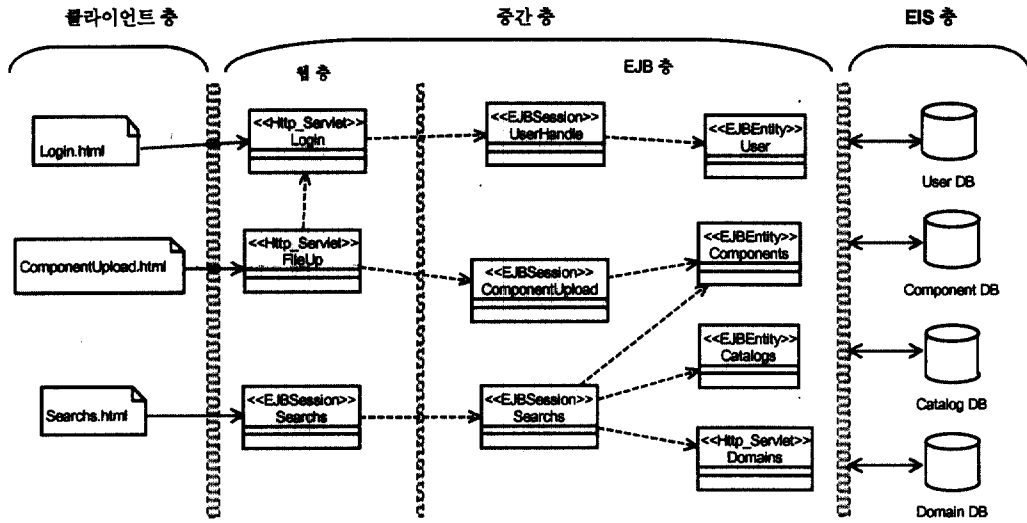



그림 8. J2EE 어플리케이션 모델 기반 컴포넌트 저장소 설계 예

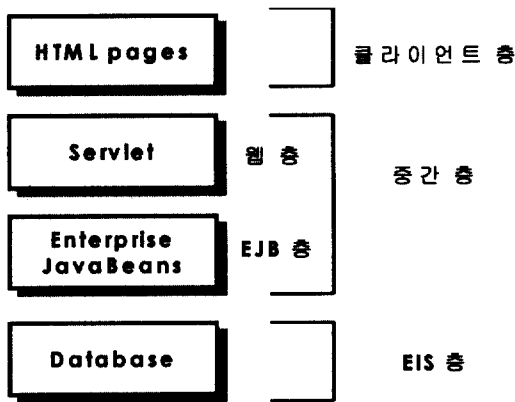



그림 9. 사용된 J2EE 요소 기술

브라우저에 동적으로 전달하도록 해주는 서버측 컴포넌트인 Servlet[2]으로 구현하였고, 비즈니스 로직을 수행하는 EJB 층은 EJB를 사용하였다. EJB는 분산 컴포넌트로서 컴포넌트 개발 모델을 제공해 주며, 또한 분산트랜잭션, 동시 사용자에 대한 제어, 멀티쓰레딩시 데이터 무결성, 보안 등의 서비스도 제공해 주므로 이러한 서비스를 비즈니스 로직 상에서 처리할 필요가 없어 어플리케이션 개발이 편리하다.

여기서는 컴포넌트 저장소의 주요 기능을 수행하는 EJB층 구현에 대해 살펴보고자 한다. EJB는 엔티티 빈과 세션 빈으로 나누어지고, 각 빈들은 컴포넌트이다. 엔티티 빈은 데이터베이스와 관련된 사항을 처리하고 세션 빈은 비즈니스 로직을 처리한다. 따라서 설계 단계에서 추출된 컴포넌트 중 데이터베이스에 관련된 Catalogs, Components, Domains, User컴포넌트는 엔티티 빈으로 구현되고, 비즈니스 로직을 수행하는 ComponentUpload, UserHandle, Search컴포넌트는 세션 빈으로 구현된다. 그리고 구현된 빈들은 EJB 서버에 배치(Deployment)되어야 동작 가능하며, 이는 배치 기술자(Deployment Descriptor)를 통하여 이루어진다.

4.1 엔티티 빈 구현 예

엔티티 빈은 데이터베이스에 저장되는 테이블을 객체화한 것으로서, 퍼시스턴스의 관리 주체에 따라 컨테이너 관리 빈과 빈 관리 빈으로 나누어진다[10]. 엔티티 빈으로 구현될 컴포넌트 중 컴포넌트, 사용자, 도메인 정보를 다루는 Components, User, Domains컴포넌트는 컨테이너 관리 빈으로, 그리고 컴포넌트 분류 정보를 다루는 Catalogs컴포넌트는 빈 관리 빈으로 구현하였다. 본 논문에서 구현한 엔티티 빈 중 컨테이너 관리 빈으로 구현된 Components컴포넌트 즉, Components엔티티 빈을 예로서 살펴보면 그림 10과 같다.

그림 10의 엔티티 빈은 컴포넌트 개발자가 컴포넌트 등록과정을 거쳐 제공한 정보를 나타내는 객체로서, 리모트 인터페이스에 해당하는 Components.java, 그리고 이를 구현한 ComponentsBean.java와 홈 인터페이스를 정의한 ComponentsHome.java, 마지막으로 컴포넌트 정보를 유일하게 식별하기 위한 pri-

```
public interface ComponentsHome extends EJBHome {
  public Components create(String com_id,
                           String    com_type,
                           String    com_content,
                           int       com_cost,
                           String    com_spec,
                           String    com_model,
                           String    com_exe,
                           Timestamp reg_date,
                           Timestamp make_date,
                           String    com_domain,
                           String    com_application)
        throws CreateException, RemoteException;
  public Components findByPrimaryKey(ComponentsPK primaryKey)
    throws FinderException, RemoteException;
  public Enumeration findByComponentType(String com_type)
    throws FinderException, RemoteException;
}
```

ComponentsHome.java

```
public interface Components extends EJBObject {
  public ComponentsInfo getComponentInfo()
    throws RemoteException;
  public String getDomain()
    throws ProcessingException, RemoteException;
  public String getApplication()
    throws ProcessingException, RemoteException;
  public void changeComponentsInfo(ComponentsInfo com_info)
    throws ProcessingException, RemoteException;
}
```

Components.java

```
public class ComponentsPK implements Serializable {
  public String com_id;
// 생략..
}
```

ComponentsPK.java

```
public class ComponentsBean implements EntityBean {
  transient EntityContext ctx = null;
  // 생략
  protected String id() {
    return (ctx == null) ? "nullctx" :
        ((ctx.getPrimaryKey() == null ? "null" :
         String.valueOf(((ComponentsPK)(ctx.getPrimaryKey())).com_id)));
  }
  public void ejbCreate(String   com_id;
                        String    com_type;
                        String    com_content;
                        int       com_cost;
                        String    com_spec;
                        String    com_model;
                        String    com_exe;
                        Timestamp reg_date;
                        Timestamp make_date;
                        String    com_domain;
                        String    com_application;)
  {
    this.com_id = com_id;
        this.com_type = com_type;
        this.com_content = com_content;
        this.com_cost = com_cost;
        this.com_spec = com_spec;
        this.com_model = com_model;
        this.com_exe = com_exe;
        this.reg_date = reg_date;
        this.make_date = make_date;
        this.com_domain =   com_domain;
        this.com_application = com_application;

  }
public void ejbActivate() {
    System.out.println("<ComponentsBean>ejbActivate called");
  }
  // 생략..
}
```

ComponentsBean.java

그림 10. Components 엔티티 빈 구현

mary key를 정의한 ComponentsPK.java로 구성된다.

EJB에서는 데이터베이스에 영구적으로 저장되는 정보를 복잡한 SQL문의 처리 없이 엔티티 빈만을 사용함으로써 진정한 객체 지향적 패러다임을 따를 수 있다. 그림 11은 SQL로 처리하는 기존의 방식과 EJB의 엔티티 빈을 사용한 경우를 나타낸 것으로서, 동일한 메소드를 구현하는 프로그래밍 방식을 비교한 것이다.

그림 11의 (a)는 특정 도메인의 컴포넌트 정보를 가져오는 getDomainComponents() 메소드를 기존의 데이터베이스 처리 방식인 SQL문을 사용하였고, (b)는 EJB의 엔티티 빈을 사용한 경우로서, SQL문의 처리 없이 엔티티 빈만을 사용함으로써 진정한 객체 지향적 패러다임을 따를 수 있다.

EJB에서는 엔티티 빈과 데이터베이스의 연결을 위해 EJB 서버 Properties파일을 사용하고, 그 예는 그림 12와 같다.

그림 12는 본 논문에서 사용한 EJB 서버인 BEA System 의 Weblogic 4.5.1에서 제공하는 properties 파일의 데이터베이스 연결 정보를 서술한 것이다. 데이터베이스 연결 정보가 EJB 서버의 properties 파일에 존재하기 때문에 이 파일만을 수정함으로써 소스 레벨의 수정이 없이 재사용이 가능하다. 따라서 데이터베이스 처리에 있어서 EJB의 엔티티 빈을 사용하게 되면 재사용성이 향상될 수 있다.

4.2 세션 빈 구현 예

세션 빈은 클라이언트 어플리케이션의 확장으로, 비즈니스 로직을 처리하는 역할을 하고 상태 저장의 유무에 따라 상태 유지 세션 빈과 무상태 세션 빈으로 나누어 진다[2,10]. 본 논문에서는 총 5개의 세션 빈이 구현되었다. 이 중 컴포넌트 등록, 사용자 등록, 사용자 로긴, 그리고 컴포넌트 검색의 경우에 사용되어지는 4개의 세션 빈은 클라이언트 정보를 유지하지 않으므로 무상태 세션 빈으로써 구현하였고, 장바구니 기능을 하는 1개의 세션 빈은 컴포넌트 구입 시 일시적인 사용자 상태 정보를 유지해야 하므로 상태 유지 세션 빈으로 구현하였다. 따라서 앞서 설명한 그림 7의 컴포넌트 추출 과정에서 추출된 컴포넌트 중 ComponentUpload, UserHandle, Search컴포넌트는 세션 빈으로서, 컴포넌트 등록, 사용자 등록 및 로긴, 컴포넌트 검색의 기능을 담당하므로 무상태 세션 빈으로 구현되어졌다. 그림 13은 무상태 세션 빈으로

```
public Vector getDomainComponents(String domain, String type)
{
            Statement stmt;
            String sqlString;
            ResultSet rs = null;

            Vector ComList = new Vector();

            sqlString = "select * from ComRegForm where comdomain ='" + domain + "' and comtype ='" +
type
                        + "' and comapplication = 'null'";

            try
                        stmt = m_connection.createStatement();
                        rs = stmt.executeQuery(sqlString);

                        while(rs.next())

                        ComponentInfo com = new ComponentInfo();

                        com.com_id = rs.getString("comid");
                        com.com_type = rs.getString("comtype");
                        com.com_cost = rs.getString("comcost");
                        com.com_content = rs.getString("comcontent");
                        com.com_spec = rs.getString("comspec");
                        com.com_model = rs.getString("commodel");
                        com.com_exe = rs.getString("comexe");
                        com.reg_date = rs.getTimestamp("comregdate");
                        com.make_date = rs.getTimestamp("commakedate");
                        com.com_domain = rs.getString("comdomain");
                        com.com_application = rs.getString("comapplication");
                        ComList.addElement(com);
                        rs.close();
                        stmt.close();
            catch(Exception ex) ex.printStackTrace();

            return ComList;
}
```

그림 11. (a) 기존의 SQL을 사용한 예

```
public Vector getDomainComponents(String domain, String type) throws ProcessingException
    {
        Context ctx = null;
        Components com = null;
        Vector v = new Vector();

        try     {
            // 엔티티빈을 얻는다.
            ctx = WLJndi.getInitialContext();
            ComponentsHome home = (ComponentsHome)ctx.lookup("repository.ejb.entity.ComponentsHome");
            Enumeration item = (Enumeration)home.findByDomain(domain);
            try {Thread.sleep(10); }catch(Exception e){}
            if(verbose) System.out.println("도메인 컴포넌트 얻음");

            // 해당 도메인 컴포넌트 정보를 얻는다.
            while(item.hasMoreElements())
            {
                    ComponentInfo ci = null;
                    com = (Components)item.nextElement();
                    ci = (ComponentInfo)com.getComponent();
                    if(ci.com_type.equals(type) && ci.com_application.equals("null"))
                            v.addElement(ci);
            }

        }catch(Exception e)     {
            throw new ProcessingException("도메인 컴포넌트 얻는데 실패하였습니다.");
        }
        finally     {
            try { if(ctx != null) ctx.close(); }catch(Exception e) {}
            return v;
        }
    }
```

그림 11. (b) 엔티티 빈을 사용한 예

```
##################################
# WEBLOGIC DEMO CONNECTION POOL PROPERTIES
# -----------------------------------------------
# CLUSTER USERS: Note that ALL JDBC connection pools should be set up
# in the *per-cluster* properties file ONLY.
#
# This connection pool uses the sample Cloudscape database shipped
# with WebLogic. Used by the EJBean, JHTML, JSP and JMS examples.
# Uncomment to use:
 weblogic.jdbc.connectionPool.connPool=₩
        url=jdbc:cloudscape:conn,₩
        driver=COM.cloudscape.core.JDBCDriver,₩
        initialCapacity=1,₩
        maxCapacity=2,₩
        capacityIncrement=1,₩
        props=user=none;password=none;server=none

# Add an ACL for the connection pool:
  weblogic.allow.reserve.weblogic.jdbc.connectionPool.connPool=guest
```

그림 12. EJB 서버 Properties파일을 통한 데이터베이스 연결의 예

```
public Components getComponent(long pid) throws Exception
    {
        Context ctx = null;
        Components com = null;

        try     {
            // 엔티티빈을 얻는다.
            ComponentsPK compk = new ComponentsPK(com_id);
            ctx = WLJndi.getInitialContext();
            ComponentsHome home = (ComponentsHome)ctx.lookup("repository.ejb.entity.ComponentsHome");
            Components item = (Components)home.findByPrimaryKey(compk);

            // 해당 id를 가진 컴포넌트 정보를 얻어낸다.
            com = item.getComponent();

            // 생략

        }catch(Exception e)     {
            throw new Exception("에러발생");
        }
        finally     {
            try { if(ctx != null) ctx.close(); }catch(Exception e) {}
            return com;
        }
    }
```

그림 13. (a) 세션 빈의 메소드 예

```
Context ctx = null;
 SearchsHome home = null;
 Searchs s = null;
 ComponentInfo com = null;

 //Search 세션빈을 얻는다.
        ctx = WLJndi.getInitialContext();
        home = (SearchsHome)ctx.lookup("repository.ejb.stateless.SearchsHome");
        s = (Searchs)home.create();

        com = s.getComponent(com_pid);
```

그림 13. (b) 세션 빈을 호출하는 클라이언트의 예

구현된 Search컴포넌트 즉, Search세션 빈의 예이다.

그림 13의 (a)는 컴포넌트 검색을 위해 만들어진 Search세션 빈의 getComponent()를 구현한 것이고, 그림 13의 (b)는 클라이언트가 Search세션 빈의 홈 인터페이스를 통하여 리모트 인터페이스를 획득한 뒤, 컴포넌트 등록 아이디를 통하여 컴포넌트 정보를 얻는 과정을 나타낸 것이다.

4.3 엔티티 빈과 세션 빈의 배치

구현된 EJB 빈들은 EJB 서버에 배치되어야 한다. 그리고 각각의 빈마다 EJB의 구조 정보와 함께 어플리케이션 조합정보에 대한 내용을 서술해야 한다. 이는 배치 기술자를 통하여 이루어지는데, EJB 스팩 1.0[17]에서는 텍스트 파일을 이용하여 기술하고, EJB 스팩 1.1에서는 XML을 이용하여 기술한다. 본 논문에서는 EJB 스팩 1.0을 따르고 있는 BEA System의 WebLogic 4.5.1을 EJB 서버로 사용하였기 때문에 텍스트 파일로 기술하였고, 그림 14와 같다.

그림 14는 배치 기술자의 예로서, EJB에서 제공해주는 서비스 중 컴포넌트 저장소 구현 시에 사용되어진 네이밍, 동시성, 트랜잭션, 보안, 결합 허용 및 데이터 맵핑 서비스 정보 등을 서술한 것이다. 이 중 대표적인 서비스로는 웹 기반의 컴포넌트 저장소에서 신뢰성 있는 트랜잭션을 제공해 주기 위해 사용한 트랜잭션 서비스가 있다. EJB에서 정의된 트랜잭션의 속성을 살펴보면 표 2와 같다.

표 2와 같이 EJB에서 정의한 런타임 트랜잭션 속성은 트랜잭션 처리 시 EJB 서버에서 제공하는 트랜잭션 서비스를 기반으로 하여 선택함으로써 묵시적으로 관리를 할 수 있다. 따라서 복잡한 트랜잭션에 관련된 프로그래밍을 하지 않고 어플리케이션 개발이 가능하게 된다. 본 논문에서는 빈이나 메소드가 오직 트랜잭션 범위 내에서만 사용될 수 있게 하는 TX_REQUIRED를 사용하였다. 웹 기반의 컴포넌트 저장소에서는 사용자의 작업 진행 중 실패할 경우 이전의 작업도 모두 무효화시키고, 작업했던 흔적을 남기지 않기 때문에 이 속성을 사용하였다. 이와 같이 배치 기술자를 사용하여 EJB에서 제공해주는 트랜잭션 등과 같은 복잡한 서비스를 직접 구현하지 않게 지원해주기 때문에 어플리케이션 개발의 효율성을 제공해주고, 또한 신뢰성 있는 서비스를 보장해준다.

그리고 많은 사용자가 접속 시에도 안정적인 서비스를 제공하기 위해서는 일반적으로 여러 개의 서버를 클러스터링으로 구성하여 시스템을 구축한다. 이때 작업을 균등 분배하기 위한 정책이 필요한데 이는 EJB 서버가 제공하는 서비스를 재사용함으로써 어플리케이션 개발자는 직접 코딩을 하지 않고 시스템을 구축할 수 있다.

그림 15는 EJB서버에서 제공하는 부하 균등화 서비스를 properties 파일을 통하여 설정한 경우를 보여주는 것으로서, 라운드 로빈 방식으로 각각의 서버에게 작업을 할당함으로써 많은 사용자가 접속 시에

그림 14. 배치 기술자 사용의 예

표 7. EJB에서 정의된 트랜잭션의 속성 [10]

트랜잭션 속성	기능
TX_NOT_SUPPORTED	빈상의 메소드를 호출하면, 그 메소드가 끝날 때까지 트랜잭션을 멈춘다.
TX_SUPPORTS	빈 혹은 메소드가 트랜잭션 내에서 호출이 되면, 트랜잭션의 범위에 포함된다
TX_REQUIRED	빈 메소드가 반드시 트랜잭션의 범위 내에서 호출되어야 한다
TX_REQUIRES_NEW	항상 새로운 트랜잭션이 시작된다
TX_MANDATORY	빈 혹은 메소드가 반드시 호출한 클라이언트의 트랜잭션 범위의 일부가 되어야 한다.
TX_BEAN_MANAGED	EJB서버에 의해서 관리되는 자체의 트랜잭션 환경을 갖지 않는다. 동일한 빈 내의 다른 메소드들은 서로 다른 트랜잭션 속성을 가질 수 있다.

```
################################
# CLUSTER-SPECIFIC PROPERTIES
# ------------------------------------------
# Cluster-specific properties in this section are set to system defaults.
# CLUSTER USERS: Note that ALL Cluster-specific properties should be set
# in the *per-cluster* properties file ONLY.
#
# Time-to-live (number of hops) for the cluster's multicast messages
# (default 1, range 1-255).
#weblogic.cluster.multicastTTL=1
#
# Sets the load-balancing algorithm to be used between
# replicated services if none is specified. If not specified,
# round-robin is used.
weblogic.cluster.defaultLoadAlgorithm=round-robin
```

그림 15. properties파일을 통한 부하 균등화 서비스의 예

도 안정적인 서비스를 제공해준다.

따라서 본 논문에서 구현된 웹 기반의 컴포넌트 저장소는 사용하고자 하는 서비스의 어려운 코딩 없이 개발이 가능하였고, 제공되는 서비스의 사용으로 신뢰성 있는 트랜잭션 서비스와 많은 사용자의 접속에도 안정적인 서비스를 보장한다.

4.4 웹 기반의 컴포넌트 저장소 실행 예

그림 16은 본 논문에서 구현한 웹 기반의 컴포넌트 저장소의 컴포넌트 등록 서비스 예를 보여준다. 컴포넌트 등록 서비스는 컴포넌트 개발자가 사용자 인증을 거친 후에 컴포넌트 등록 폼을 완성함으로써 컴포넌트를 등록한다. 등록 폼의 내용으로는 컴포넌트 이름, 컴포넌트 도메인 분류 정보, 컴포넌트 가격, 컴포넌트 요약 정보, 컴포넌트 제조 날짜, 컴포넌트 명세서, 컴포넌트 모델링 정보 및 컴포넌트 실행 파일이 있다.

5. 결론 및 향후 연구 과제

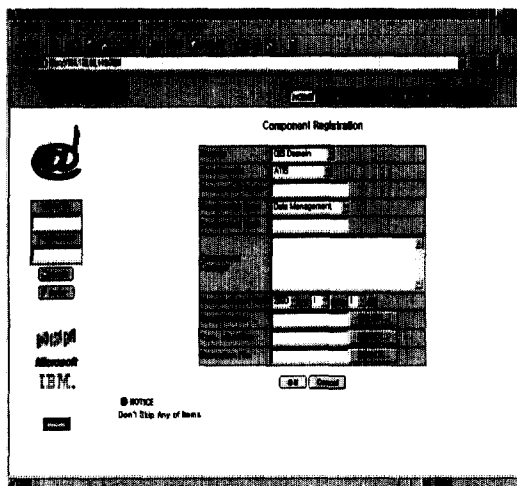
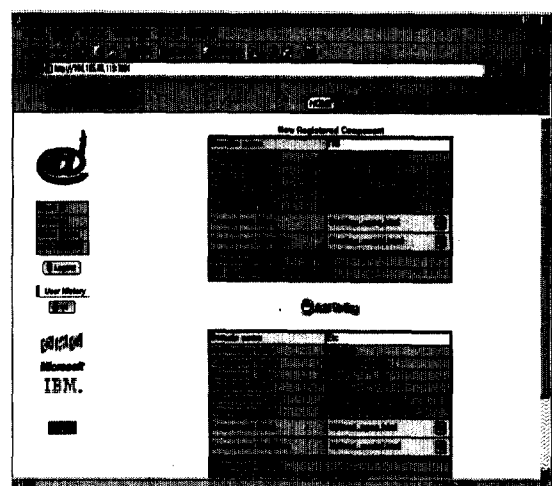
웹 기반의 컴포넌트 저장소는 컴포넌트의 등록, 저장, 획득, 유지 관리 등의 기능성과 함께 트랜잭션 처리에 있어서의 신뢰성과 안정성을 보장하고 다수 사용자에 의한 동시 접근을 안정적으로 지원해야 한다. 또한 컴포넌트의 변동 및 관리 정책의 변화에 유연성 있게 대처하고, 컴포넌트 저장소 자체의 기능 확장을 효과적으로 지원하여야 한다. 이러한 기능적, 비기능적인 요구사항을 만족시키기 위하여 웹 기반 어플리케이션 개발에 적합한 모델로 인식되고 있는 J2EE를 바탕으로 웹 기반 컴포넌트 저장소를 구현하였다. J2EE 환경이 제공해주는 확장성, 신뢰성, 안정성, 동시성 등에서의 이점을 웹 기반의 컴포넌트 저장소 개발에 효과적으로 반영하기 위하여, J2EE 어플리케이션 모델을 고려하여 분산 컴포넌트 기반으로 컴포넌트 저장소를 분석 및 설계하였다. 따라서 개발된 웹 기반 컴포넌트 저장소에서는 J2EE 환경이 지원하는 확장성, 신뢰성, 안정성, 동시성 등의 비기능성이 만족됨을 기대할 수 있다. 또한 컴포넌트 형태로 저장소를 분석, 설계, 구현함으로써 컴포넌트 저장소 기능의 확장성, 이식성, 그리고 유지 관리의 효율성을 기대할 수 있다. 특히 J2EE 어플리케이션 모델 중 비즈니스 로직 부분을 담당하는 EJB층의 구현 시에

(a) 컴포넌트 등록 폼

(b) 등록된 컴포넌트의 예

그림 16. 컴포넌트 등록 서비스의 예

컴포넌트 모델인 EJB를 사용함으로써 데이터 액세스, 동시성, 트랜잭션 등과 같은 문제를 복잡한 코딩 없이 단지 배치 기술자만을 서술하여 어플리케이션 서버(EJB 서버)에서 제공하는 서비스를 이용하여 해결하였다. 그리고 부하 균등 서비스를 사용하여 웹 어플리케이션의 문제점인 다수 사용자에 의한 서비스 불안정 문제를 해결할 수 있었다.

향후 연구 과제로는 컴포넌트 저장소에서 제공하는 컴포넌트에 대한 명세 방법과 컴포넌트 분류 기법에 대한 연구가 필요하다. 컴포넌트 명세는 요구사항에 적합한 컴포넌트를 결정하는데 유용하고, 컴포넌트 분류는 컴포넌트 유지 관리 및 기능성과 비기능성이 함께 고려된 컴포넌트를 추출하는데 유용하게 활용될 수 있다.

참 고 문 헌

[1] Sun Microsystems, Inc., "Java™ 2 Platform Enterprise Edition Specification", ver 1.2, 1999.

[2] 김세곤, 서창근, 김민식, EJB Bible, 정보문화사, 서울, 2001.

[3] 최유희, 염근혁, "분산 컴포넌트 기반의 소프트웨어 설계 방법", 제 27회 한국정보과학회 추계 학술 발표 논문집, 제 27권 2호(I), pp.498-500, 2000.

[4] 최유희, "분산 컴포넌트 기반의 소프트웨어 분석 및 설계 방법", 석사 학위 논문, 2001.

[5] OMG, "OMG Unified Modeling Language Specification", ver 1.3, March 1999.

[6] R., Seacord, S., Hissam and K., Wallnau, "Agora: A Search Engine for Software Components", IEEE Internet Computing, pp.62-70, 1998.

[7] ComponentSource, http://www.componentsource.com/.

[8] Imagicom, http://www.imagicom.com/.

[9] 김창호, Enterprise JAVA BEANS, 도서출판 대림, 서울, 2001.

[10] R., Monson-Haefel, Enterprise JAVA BEANS, O'Reilly & Associates, 2000.

[11] N., Kassem and the Enterprise Team, Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition, Addision-Wesley, 2000.

[12] E., Roman, Mastering Enterprise JavaBeans™ and the Java™ 2 Platform, Enterprise Edition, Addision-Wesley, 1999.

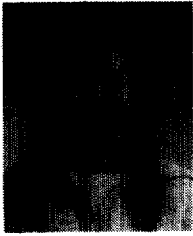
[13] 파워빌더 뉴스, http://powersoft.penta.co.kr/resources/pb/news/pbnews.htm/, 2000년 1월.

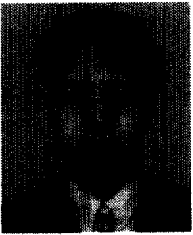
[14] Sun Microsystems, Inc., http://www.javastudy.co.kr/docs/jhan/j2ee/overview3.html/.

[15] Sun Microsystems, http://www.javastudy.co.kr/docs/jhan/j2ee/blueprints.html/.

[16] V., Matena and B., Stearns, Applying Enterprise JavaBeans:Component-Based Development for the J2EE™ Platform, Addision-Wesley, 2001.

[17] Sun Microsystems, Inc., "Enterprise JavaBeans Specification 1.0", http://java.sun.com/.

[18] G., Booch, J., Rumbaugh, and I., Jacobson, The Unified Modeling Language User Guide, Addision-Wesley, 1999.

안 성 아

1999년 2월 부산대학교 컴퓨터공학과(공학사)
2001년 3월~현재 부산대학교 컴퓨터공학과 석사과정
관심분야 : 컴포넌트 저장소, 컴포넌트 명세 및 분류 기법, 컴포넌트 기반 소프트웨어 개발 방법론, 분산 컴포넌트, 소프트웨어 재사용 등임.

최 희 석

1998년 2월 부산대학교 컴퓨터공학과(공학사)
2000년 2월 부산대학교 컴퓨터공학과(공학석사)
2000년 3월~현재 부산대학교 컴퓨터공학과 박사과정
관심분야 : 컴포넌트 기반 소프트웨어 개발 방법론, 소프트웨어 아키텍쳐, 분산 컴포넌트, 컴포넌트 저장소, 도메인 공학, 소프트웨어 재사용 등임.

염 근 혁

1985년 서울대학교 계산통계학과(학사)
1992년 Univ. of Florida 전산학과(석사)
1995년 Univ. of Florida 전산학과(박사)
1985년 1월~1988년 2월 금성반도체 컴퓨터연구실 연구원
1988년 3월~1990년 6월 금성사 정보기기연구소 주임연구원
1995년 9월~1996년 8월 삼성SDS 정보기술연구소 책임연구원
1996년 8월~현재 부산대학교 컴퓨터공학과 조교수
부산대학교 컴퓨터 및 정보통신 연구소 연구원
관심분야 : 컴포넌트 기반 소프트웨어 개발, 도메인 공학, 소프트웨어 아키텍쳐, 객체지향 소프트웨어 개발방법론, 요구 검증, 분산 컴포넌트, 소프트웨어 재사용 등임.